| L Number | Hits | Search Text | DB | Time stamp |
|---|---|---|---|---|
| 1 | 3639 | 345/87-103.ccls. | USPAT | 2003/09/10 15:41 |
| 3 | 4 | 345/87-103.ccls. and ((ferroelectric or ferro-electric or flc) WITH (polarit$3 or positive) WITH asymmetric$3) | USPAT | 2003/09/10 15:43 |
| 2 | 14 | (ferroelectric or ferro-electric or flc) WITH (polarit$3 or positive) WITH asymmetric$3 | USPAT | 2003/09/10 16:02 |
| 4 | 15 | (ferroelectric or ferro-electric or flc) WITH asymmetric$3 and (polareit$3 or positive or opposit$3) WITH asymmetric$3 | USPAT | 2003/09/10 15:55 |
| 5 | 9 | ((ferroelectric or ferro-electric or flc) WITH asymmetric$3 and (polareit$3 or positive or opposit$3) WITH asymmetric$3) not ((ferroelectric or ferro-electric or flc) WITH (polarit$3 or positive) WITH asymmetric$3) | USPAT | 2003/09/10 15:55 |
| 6 | 9 | ((ferroelectric or ferro-electric or flc) WITH asymmetric$3 and (polareit$3 or positive or opposit$3) WITH asymmetric$3) not ((ferroelectric or ferro-electric or flc) WITH (polarit$3 or positive) WITH asymmetric$3) | USPAT | 2003/09/10 16:01 |
| 7 | 266 | 345/96.ccls. | USPAT | 2003/09/10 16:02 |
| 8 | 138 | 345/210.ccls. | USPAT | 2003/09/10 16:02 |
| 9 | 12408 | (ferroelectric or ferro-electric or flc) | USPAT | 2003/09/10 16:02 |
| 10 | 389 | 345/96.ccls. or 345/210.ccls. | USPAT | 2003/09/10 16:02 |
| 11 | 82 | ((ferroelectric or ferro-electric or flc) ) and (345/96.ccls. or 345/210.ccls.) | USPAT | 2003/09/10 16:02 |
| 12 | 79 | (((ferroelectric or ferro-electric or flc) ) and (345/96.ccls. or 345/210.ccls.)) and (more or three or fields) | USPAT | 2003/09/10 16:16 |
| 13 | 1 | (((ferroelectric or ferro-electric or flc) ) and (345/96.ccls. or 345/210.ccls.)) and (more or three) NEAR3 fields | USPAT | 2003/09/10 16:03 |
| 14 | 177 | (345/96.ccls. or 345/210.ccls.) and (fields or frames) | USPAT | 2003/09/10 16:17 |
| 15 | 137 | ((345/96.ccls. or 345/210.ccls.) and (fields or frames)) not ((((ferroelectric or ferro-electric or flc) ) and (345/96.ccls. or 345/210.ccls.)) and (more or three or fields)) | USPAT | 2003/09/10 16:40 |

# CACHE ORGANIZATION— DIRECT MAPPED CACHE

## BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention relates to the field of computer graphics and caching, and, more specifically, to the caching of computer graphics elements.



2. Background Art

Computers are often used to display graphical information. In some instances, graphical data or images are "rendered" by executing instructions from an application that is drawing the data or image to a display. An image is a regular two dimensional array in which every element of the array is a digital quantity of energy such as light/heat/density, etc. An image may also be viewed as a discretely sampled digital representation of an energy continuum with the same number of elements in each row. The image can also be procedurally generated dynamically at the time of display by the computer program or sampling device, for example. A displayed image may be made up of a plurality of graphical objects. Examples of graphical objects include points, lines, polygons, and three dimensional solid objects.

If you looked closely at a television screen, computer display, magazine page, etc., you would see that an image is made up of hundreds or thousands of tiny dots, where each dot is a different color. These dots are known as picture elements, or "pixels" for short when they are on a computer display and as dots when printed on a page. The color of each pixel is represented by a number value. To store an image in a computer memory, the number value of each pixel of the picture is stored. The number value represents the color and intensity of the pixel.

The accuracy with which a document can be reproduced is dependent on the "resolution" of the pixels that make up the document. The resolution of a pixel is the size of the number value used to describe that pixel. The size of the number value is limited by the number of "bits" in the memory available to describe each pixel (a bit is a binary number having a value of 1 or 0). The greater the number of bits available per pixel, the greater the resolution of the document. For example, when only one bit per pixel is available for storage, only two values are available for the pixel. If two bits are available, four levels of color or intensity are available. While greater resolution is desirable, it can lead to greater use of data storage. For example, if each pixel is represented by a 32-bit binary number, 320,000 bits of information would be required to represent a 100×100 pixel image. Such information is stored in what is referred to as a "Frame Buffer" (or "G array").

The process of converting graphics data and instructions into a display image is known as "pixel rendering." During pixel rendering, color and other details can be applied to areas and surfaces of these objects using "texture mapping" techniques. In texture mapping, a texture image (also referred to as a texture map, or simply as a texture) is



mapped to an area or surface of a graphical object to produce a visually modified object with the added detail of the texture image. A texture image may contain, for example, an array of RGB (red, green, blue) color values, intensity values, or opacity values.

As an example of texture mapping, given a featureless graphical object in the form of a cube and a texture image defining a wood grain pattern, the wood grain pattern of the texture image may be mapped onto one or more surfaces of the cube such that the cube appears to be made out of wood. Other examples of texture mapping include mapping of product logo texture images to computer-modeled products, or mapping of texture images containing vegetation and trees to a barren computer-modeled landscape. Textures mapped onto geometric surfaces may also be used to provide additional motion and spatial cues that surface shading alone might not be capable of providing. For example, a featureless sphere rotating about an axis appears static until an irregular texture image or pattern is mapped to its surface.

Texture mapping involves using a texture image having a function defined in texture space. Typically, the texture space is represented as a two dimensional space, with "S" and "T" indices defining orthogonal axes (e.g., horizontal and vertical). A texture image is represented in texture space as an array in S and T of discrete texture elements or values called "texels." The texture image is warped or mapped from the texture space into an image space having an array of picture elements called "pixels." The pixels are associated with orthogonal axis coordinates "X" and "Y" in the image space which define a viewing plane for display. Based on the particular mapping function, a correspondence is generated between pixels representing an object or primitive in the image space and texels representing a texture image in the texture space.

Typically, a two-dimensional texture or pattern image is mapped onto a two or three-dimensional surface. For a two-dimensional surface, X and Y coordinates may be sufficient for defining a mapping function between pixels forming the surface and texels forming the texture image. For a three-dimensional surface, a perspective coordinate or other depth cueing mechanism may be provided to indicate distance from the viewing plane defined by the X and Y axes. The perspective coordinate may then be applied to the mapping function. For example, as the perspective coordinate value for a surface region increases (i.e., the surface region is further from the viewing plane), the mapping of the texture image may be darkened and/or compressed (i.e., neighboring pixels in the surface region will span an increased number of texels in the texture image), or otherwise warped, relative to surface regions having a lower perspective coordinate value. Through the application of depth cueing, the viewer is provided with a sense of distance or depth when viewing the rendered pixels.

FIGS. 1A–1C illustrate a mapping of a brick-pattern texture image in texture space to a triangle primitive (**100**) in image space. FIG. 1A illustrates triangle primitive **100** in image space prior to texture mapping. FIG. 1B illustrates the brick-pattern texture image in texture space. FIG. 1C illustrates triangle primitive **100** in image space after texture mapping has completed.

In FIG. 1A, triangle primitive **100** is defined by vertices at X,Y coordinate pixel locations $P_A(X_A,Y_A)$, $P_B(X_B,Y_B)$ and $P_C(X_C,Y_C)$, where X is the horizontal axis and Y is the vertical axis. Pixels defining the perimeter of triangle primitive **100** may be explicitly stored in memory, or, to reduce storage requirements for individual primitives, the perimeter

| L Number | Hits | Search Text | DB | Time stamp |
|---|---|---|---|---|
| 1 | 3639 | 345/87-103.ccls. | USPAT | 2003/09/10 15:41 |
| 3 | 4 | 345/87-103.ccls. and ((ferroelectric or ferro-electric or flc) WITH (polarit$3 or positive) WITH asymmetric$3) | USPAT | 2003/09/10 15:43 |
| 2 | 14 | (ferroelectric or ferro-electric or flc) WITH (polarit$3 or positive) WITH asymmetric$3 | USPAT | 2003/09/10 16:02 |
| 5 | 9 | ((ferroelectric or ferro-electric or flc) WITH asymmetric$3 and (polareit$3 or positive or opposit$3) WITH asymmetric$3) not ((ferroelectric or ferro-electric or flc) WITH (polarit$3 or positive) WITH asymmetric$3) | USPAT | 2003/09/10 15:55 |
| 6 | 9 | ((ferroelectric or ferro-electric or flc) WITH asymmetric$3 and (polareit$3 or positive or opposit$3) WITH asymmetric$3) not ((ferroelectric or ferro-electric or flc) WITH (polarit$3 or positive) WITH asymmetric$3) | USPAT | 2003/09/10 16:01 |
| 7 | 266 | 345/96.ccls. | USPAT | 2003/09/10 16:02 |
| 8 | 138 | 345/210.ccls. | USPAT | 2003/09/10 16:02 |
| 9 | 12408 | (ferroelectric or ferro-electric or flc) | USPAT | 2003/09/10 16:02 |
| 10 | 389 | 345/96.ccls. or 345/210.ccls. | USPAT | 2003/09/10 16:02 |
| 12 | 79 | (((ferroelectric or ferro-electric or flc) ) and (345/96.ccls. or 345/210.ccls.)) and (more or three or fields) | USPAT | 2003/09/10 16:16 |
| 13 | 1 | (((ferroelectric or ferro-electric or flc) ) and (345/96.ccls. or 345/210.ccls.)) and (more or three) NEAR3 fields | USPAT | 2003/09/10 16:03 |
| 14 | 177 | (345/96.ccls. or 345/210.ccls.) and (fields or frames) | USPAT | 2003/09/10 16:17 |
| 15 | 137 | ((345/96.ccls. or 345/210.ccls.) and (fields or frames)) not ((((ferroelectric or ferro-electric or flc) ) and (345/96.ccls. or 345/210.ccls.)) and (more or three or fields)) | USPAT | 2003/09/10 17:38 |
| 16 | 7 | Saishu-$.in. | US-PGPUB | 2003/09/10 17:16 |
| 17 | 1 | Saishu-$.in. and 345/97.ccls. | US-PGPUB | 2003/09/10 17:18 |
| 18 | 1 | (Saishu-$.in. and 345/97.ccls.) and three NEAR3 fields | US-PGPUB | 2003/09/10 17:18 |
| 4 | 15 | (ferroelectric or ferro-electric or flc) WITH asymmetric$3 and (polareit$3 or positive or opposit$3) WITH asymmetric$3 | USPAT | 2003/09/10 17:39 |
| 11 | 82 | ((ferroelectric or ferro-electric or flc) ) and (345/96.ccls. or 345/210.ccls.) | USPAT | 2003/09/10 17:29 |
| 19 | 708 | 345/87-103.ccls. and ((ferroelectric or ferro-electric or flc) ) | USPAT | 2003/09/10 17:38 |
| 20 | 40814 | asymmetric$3 and (polareit$3 or positive or opposit$3) | USPAT | 2003/09/10 17:40 |
| 21 | 48 | (345/87-103.ccls. and ((ferroelectric or ferro-electric or flc) )) and ( asymmetric$3 and (polareit$3 or positive or opposit$3) ) | USPAT | 2003/09/10 17:40 |
| 23 | 1118735 | "48" not ((ferroelectric or ferro-electric or flc) WITH (polarit$3 or positive) WITH asymmetric$3) | USPAT | 2003/09/10 17:41 |

pixels may be linearly interpolated from the vertices defined by pixel locations $P_A(X_A,Y_A)$, $P_B(X_B,Y_B)$ and $P_C(X_C,Y_C)$. The interior of triangle primitive **100** is formed by those pixel locations that lie within the defined perimeter. The surface of triangle primitive **100** in this example comprises the union of the perimeter pixels (including the vertices) and the interior pixels.

In FIG. 1B, a brick pattern is stored as a texture image referenced by S and T coordinates. The brick pattern may be mapped to pixels in image space by accessing texels at integer S and T coordinates. In accordance with a particular mapping function, pixel vertices $P_A$, $P_B$ and $P_C$ of the triangle primitive correspond to S and T coordinates $(S_A,T_A)$, $(S_B,T_B)$ and $(S_C,T_C)$, respectively. The orientation of the mapped vertices indicates rotation and scaling of triangle primitive **100** with respect to the S and T texture space.

FIG. 1C shows triangle primitive **100** having pixel vertices with corresponding X,Y coordinates for the image space, as well as texture space S,T coordinates for extracting texel values. The pixel vertices are $P_A(X_A,Y_A; S_A,T_A)$, $P_B(X_B,Y_B; S_B,T_B)$ and $P_C(X_C,Y_C; S_C,T_C)$. The brick pattern of the texture image of FIG. 1B appears within triangle primitive **100** at slightly reduced scale, and at an approximately forty-five degree rotational offset from the X-axis. Other texture images may be similarly texture mapped to surfaces in render operations.

As the pixels defining a surface are rendered, S and T coordinate values are generated for each pixel based on the mapping function. The generated S and T coordinate values are then used to obtain a texel value for each rendered pixel in the image space. However, the generated S and T coordinate values are generally fractional values (i.e., not integer values). Consequently, the generated S and T coordinate values often correspond to a location in the texture space that falls between the texels of the texture image array.

Several options exist for selecting a texture value, given real S and T coordinate values. One of the simplest options is to round the S and T coordinate values to the nearest integers, and then select the texel corresponding to the rounded integer coordinate values. A more accurate representation is produced by interpolating between the four nearest samples that surround the real (S,T) location. For example, a bilinear interpolation algorithm (i.e., bilinear filtering), or higher-order interpolation algorithm, may be used to interpolate texel values for fractional S and T coordinates. Bilinear interpolation is illustrated in FIG. 2.

In FIG. 2, a pixel $P_N$ is mapped to S and T coordinates $(L+\alpha,M+\beta)$. The four nearest texels in texture space are TXL(L,M), TXL(L+1,M), TXL(L,M+1) and TXL(L+1,M+1). To perform bilinear interpolation (or filtering), a linear interpolation is performed between the texel pairs [TXL(L,M), TXL(L+1,M)] and [TXL(L,M+1), TXL(L+1,M+1)] to determine intermediate pixel values $P_N'$ $(L+\alpha,M)$ and $P_{N''}$ $(L+\beta,M+1)$, respectively. These linear interpolation functions are performed to implement equations (1) and (2) below.

$$P'_N(L+\alpha,M)=(1-\alpha)TXL(L,M)+\alpha TXL(L+1,M) \quad (1)$$

$$P''_N(L+\alpha,M+1)=(1-\alpha)TXL(L,M+1)+\alpha TXL(L+1,M+1) \quad (2)$$

A third linear interpolation operation is performed on intermediate pixel values $P'_N(L+\alpha,M)$ and $P''_N(L+\alpha,M+1)$ to obtain $P_N(L+\alpha,M+\beta)$ in accordance with the following equation (3). The linear interpolation operations for the intermediate pixels may be performed along the opposite axis as well, or the linear interpolation operations may be combined to implement a form of equation (4) below.



$$P_N(L+\alpha,M+\beta)=(1-\beta)P'_N(L+\alpha,M)+\beta P''_N(L+\alpha,M+1) \quad (3)$$

$$=[1-(\alpha+\beta)+\alpha\beta]TXL(L,M)+[\alpha(1-\beta)]TXL(L+1,M)+[\beta(1-\alpha)]TXL(L,M+1)+\alpha\beta TXL(L+1,M+1) \quad (4)$$

The above equations (1)–(4) may be implemented in any equivalent form, or the equations may be approximated, to optimize the calculation apparatus for speed and/or complexity.

Using the texel selection processes described above, severe aliasing of the texture may occur if the surface being texture-mapped is far from the viewing plane. This aliasing is caused when the reduced pixel resolution provides insufficient sampling of texture images that have higher frequency components (e.g., fast transitioning color or intensity values). The interpolated (S,T) values may skip over large areas of the texture. A technique known as MIP-mapping is often performed to prevent aliasing by precomputing multiple, filtered copies of the texture at successively lower resolutions. For example, a texture image comprising a 256×256 texel array would be filtered and resampled to obtain further texel arrays (or maps) at 128×128, 64×64, 32×32, 16×16, 8×8, 4×4, and 2×2 resolutions. The cost of storing the additional texel arrays is an increase of approximately thirty percent in memory size.

The particular size of the texel array that is used during pixel rendering is chosen based on a computer parameter known as the "level of detail." The level of detail represents the relative distance between the interpolated S and T values. Each texel array size represents an integer level of detail, and the computed level of detail values are real numbers. High quality texture mapping is obtained by performing bilinear interpolation in the texel array representing the integer level of detail immediately above and below the computed level of detail of each pixel. Next, a linear interpolation is performed between the integer levels of detail to obtain the texture value at the non-integer level of detail. This process is known as trilinear MIP-mapping.

To facilitate texture mapping, a texture image may be stored in a dynamic random access memory (DRAM) device. The texel values of the texture image are accessed from the DRAM as needed to determine pixel values for a rendered image in the frame buffer. Unfortunately, DRAM devices are inefficient when performing data transfer operations (e.g., data reads) for individual data values. Peak efficiency is achieved when transferring multiple data values, especially data values that are in adjacent memory locations. For example, for a burst transfer of data in consecutive locations, a DRAM device may support a transfer rate of eight bytes per clock cycle. The same DRAM device may have a transfer rate of one byte per nine clock cycles for arbitrary single byte transfers. These performance characteristics are not well-suited to texture mapping.

In texture mapping operations, pixel values for a frame buffer are often determined in a particular scan order, such as by scanning in the direction of the X axis. However, texels associated with consecutive pixels are rarely in a predictable scan order with respect to texture space. For example, in the texture mapping process of FIGS. 1A–1C, a scan along the X axis in image space results in a scan pattern in the texture space that includes multiple passes from the left edge (T axis) of the texture image towards the upper right of the texture image. FIG. 3 illustrates the scan direction in texture space for the texture mapping of FIGS. 1A–1C. As shown, each scan arrow represents texel accesses that frequently traverse, or "skip", rows, including large skips between the ending of one scan arrow and the beginning of the next scan arrow based on the boundaries defined by the primitive in image space.

| | | | | |
|---|---|---|---|---|
| 24 | 44 | ((345/87-103.ccls. and ((ferroelectric or ferro-electric or flc) )) and ( asymmetric$3 and (polareit$3 or positive or opposit$3) )) not ((ferroelectric or ferro-electric or flc) WITH (polarit$3 or positive) WITH asymmetric$3) | USPAT | 2003/09/10 17:42 |

For a linearly configured DRAM, for example, because the texels in a texture image are not typically scanned in a linear path along the S axis, consecutive pixels will access texels that are widely dispersed across memory. For a 1024×1024 texture image in which each texel is one byte wide, a traversal of one integer T coordinate may translate to a skip of 1024 bytes in DRAM. These memory skips are not easily predictable because the skips are dependent upon the size of the image, the width of a texel, the rotational angle between the S,T axes and the X,Y axes, etc. Texture mapping may also be nonlinear for irregular surfaces, further dispersing memory access operations.

FIG. 4 illustrates an example pixel scan line progressing through a portion of a texel array. The texel array shown encompasses the range [L, L+5] in the S direction and [M,M+4] in the T direction. The pixels that form the scan line comprise $P_N$, $P_{N+1}$, $P_{N+2}$, $P_{N+3}$, $P_{N+4}$, $P_{N+5}$ and $P_{N+6}$. $P_N$ lies within the texel neighborhood formed by texels at (L,M+1), (L+1,M+1), (L,M+2) and (L+1,M+2). Pixel $P_{N+1}$ has a texel neighborhood of texels at (L+1,M), (L+2,M), (L+1,M+1), and (L+2,M+1). Each of pixels $P_{N+2}$, $P_{N+3}$, $P_{N+4}$, $P_{N+5}$ and $P_{N+6}$ have a similar texel neighborhood. These texels may be used to determine the nearest neighbor for approximating the desired texel value. Also, as described above, interpolation may be performed on the texel neighborhood of each pixel. Assuming a linear memory in S, and memory access of a texel neighborhood in the order of (top-left, top-right, bottom-left, bottom-right), the memory transfers for the texels associated with pixels $P_N$, $P_{N+1}$, $P_{N+2}$, $P_{N+3}$, $P_{N+4}$, $P_{N+5}$ and $P_{N+6}$ may occur as shown in the following table (where the texture image size is W(width)× H(height), and the base address "B" of the texel array is at (L,M)):

| PIXEL | TEXEL | RAM LNR ADDR | DISTANCE (SKIP) |
|---|---|---|---|
| $P_N$ | (L, M + 1) | B + W | — |
| | (L + 1, M + 1) | B + W + 1 | 1 |
| | (L, M + 2) | B + 2W | W − 1 |
| | (L + 1, M + 2) | B + 2W + 1 | 1 |
| $P_{N+1}$ | (L + 1, M) | B + 1 | −2W |
| | (L + 2, M) | B + 2 | 1 |
| | (L + 1, M + 1) | B + W + 1 | W − 1 |
| | (L + 2, M + 1) | B + W + 2 | 1 |
| $P_{N+2}$ | (L + 2, M) | B + 2 | −W |
| | (L + 3, M) | B + 3 | 1 |
| | (L + 2, M + 1) | B + W + 2 | W − 1 |
| | (L + 3, M + 1) | B + W + 3 | 1 |
| $P_{N+3}$ | (L, M + 3) | B + 3W | 2W − 3 |
| | (L + 1, M + 3) | B + 3W + 1 | 1 |
| | (L, M + 4) | B + 4W | W − 1 |
| | (L + 1, M + 4) | B + 4W + 1 | 1 |
| $P_{N+4}$ | (L + 1, M + 2) | B + 2W + 1 | −2W |
| | (L + 2, M + 2) | B + 2W + 2 | 1 |
| | (L + 1, M + 3) | B + 3W + 1 | W − 1 |
| | (L + 2, M + 3) | B + 3W + 2 | 1 |
| $P_{N+5}$ | (L + 2, M + 1) | B + W + 2 | −2W |
| | (L + 3, M + 1) | B + W + 3 | 1 |
| | (L + 2, M + 2) | B + 2W + 2 | W − 1 |
| | (L + 3, M + 2) | B + 2W + 3 | 1 |
| $P_{N+6}$ | (L + 3, M + 1) | B + W + 3 | −W |
| | (L + 4, M + 1) | B + W + 4 | 1 |
| | (L + 3, M + 2) | B + 2W + 3 | W − 1 |
| | (L + 4, M + 2) | B + 2W + 4 | 1 |

Associated with each of the pixels above ($P_N$–$P_{N+6}$) is a skip in the DRAM texel access of approximately the width of the texture image which is caused by the two-dimensional nature of the texel neighborhood. Even larger skips are introduced when the scan pattern crosses multiple integer coordinates in T for consecutive pixels. The speed of the texture mapping process may be significantly reduced by the performance of DRAM data transfers with frequent address skips of this nature.

Prior art texture mapping schemes attempt to overcome the limitations of DRAM data transfer characteristics by using a smaller, faster buffer memory to hold data between DRAM transfers. Buffering consists of loading a block of contiguous data into buffer memory for use by the processor performing the texel processing. A new block of data is loaded from DRAM into the buffer when it is needed.

FIG. 5 illustrates buffering applied to a texture image. In FIG. 5, texture image **500** has dimensions W×H, and base address **502**. An N-byte buffer is used to hold N-byte buffered block **501** of texture image data having starting address M. Buffered block **501** is loaded as a linear block of texture image data from memory, or as a multidimensional tile of contiguous texture image data.

Buffering apparatus are illustrated in FIG. 6. DRAM **600** is coupled to N-byte buffer **601** to receive address information **604**, and to transfer read data block **603** into the buffer memory. The data transferred from DRAM **600** comprises DRAM locations M through M+N, where M is supplied as read address **604**. The N-byte contiguous block of data in buffer **601** is available via bus **605** for texel processing component **602** to access the texture image data in buffer **601**. The texture image data is used to produce output **606**, such as rendered pixels for display. When the texture image data required by the texel processing component **602** is not located in buffer **601**, a new contiguous buffered block of texture image data is retrieved from DRAM **600** and placed in buffer **601**.

Rather than performing the transfer of data as a single block to the buffer, the data may be streamed through the buffer, for example, in a FIFO (first in, first out) arrangement. The streaming data has an accessibility lifetime in the buffer based on the time required to shift a data element through the buffer and out. This lifetime is directly related to the size of the buffer itself.

If the buffered block **501** is configured as a contiguous one-dimensional (or linear) block of data, the buffered data is strongly biased along the S axis direction. Therefore, for two-dimensional graphics applications such as texel processing, buffered block **501** in buffer **601** requires frequent transfers from DRAM **600** to track texels when scan patterns produced by a particular mapping have a strong T component that causes frequent skips. Any performance gain achieved by storing a contiguous memory block in buffer **601** is countered by the need to make frequent data transfers of different blocks from DRAM **600** to buffer **601**. Due to the contiguous nature of a buffer, a buffer needs to be very large to encompass large skips within the buffer, particularly for large images.

In the prior art, U.S. Pat. No. 5,548,709, issued to Hannah et al. on Aug. 20, 1996, discloses a semiconductor chip, referred to as TRAM (texture random access memory), that integrates texture memory, interpolation and resampling logic on the same substrate. Textures are input to the chip and stored in a main memory. The interpolator produces an output texel by interpolating from the textures stored in memory.

## SUMMARY OF THE INVENTION

The invention provides for the cache organization of texture information and a method and apparatus for accessing the cached texture information and an index for said cached information. Texture image data (also called texels)